Correlation Analysis of MQTT Loss and Delay According to QoS Level

Shinho Lee, Hyeonwoo Kim, Dong-kweon Hong, Hongtaek Ju
Dept. of Computer Engineering
Keimyung University
Daegu, Republic of Korea
(leeshinho, hwkim84, dkhong, juht}@kmu.ac.kr

Abstract—**MQTT is an open protocol developed and released by IBM. To ensure the reliability of message transmission, MQTT supports three levels of QoS. In this paper, we analyze MQTT message transmission process which consists of real wired/wireless publish client, broker server and Subscribe client. By transmitting messages through 3 levels of QoS with various sizes of payloads, we have captured packets to analyze end-to-end delays and message loss.**

Keywords : MQTT, Broker server, Publish, Subscribe

I. INTRODUCTION

Today, due to the increased use of smartphones, push notification services[9] are now commonly used. Push notifications service keeps the device online for every certain communication cycle, and the server pushes the messages to each client when necessary. Compared to the polling method, push notification method was proved to be more efficient in battery and data consumption[7].

MQTT (Message Queue Telemetry Transport)[1], an open protocol designed by IBM, was originally intended for unreliable networks with restricted resources such as low bandwidth and high-latency. MQTT consists of one broker server and two kinds of clients that are called Publisher (Publish client) and Subscriber (Subscribe client). Broker server acts as an intermediary for messages sent between Publish client and Subscribe client for the interesting topic. When the Publish client issues a topic and sends a message to the Broker server, the Subscribe client selects the topics which it finds interesting.

As an example, Facebook Messenger[4] is based on MQTT, and many other open source projects are on the way based on MQTT. MQTT Protocol is suitable for implementing integrated SNS gateway servers which can merge different SNS protocols and OS into a unified single platform. Also, there is no restriction in messaging while using push notification services.

To ensure the reliability of messaging, MQTT supports 3 levels of Quality of Services(QoS)[8]. Figure 1 shows packet exchange measure according to 3 different QoS levels. QoS Level 0 sends message only once following the message distribution flow, and does not check whether the message arrived to its destination. Therefore, in case of sizable messages, it is possible that the message will be lost when any kind of loss comes in the way. QoS Level 1 sends the message at least once, and checks the delivery status of the message by using the status check message, PUBACK. However, when PUBACK is lost, it is possible that the server will send the same message twice, since it has no confirmation of the message being delivered. QoS Level 2 passes the message through exactly once utilizing the 4-way handshake. It is not possible to have a message loss in this level, but due to the complicated process of 4-way handshake, it is possible to have relatively longer end-to-end delays.

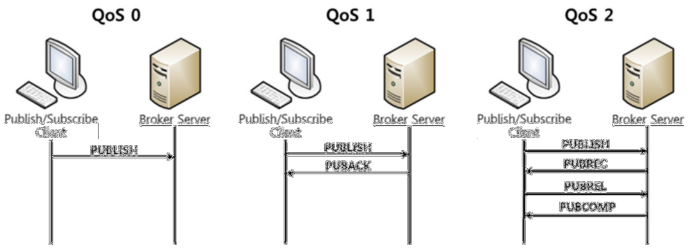

Figure 1. Packet transmission method about QoS Level

The higher QoS level is the more packets will be exchanged. It is true that higher-level QoS is more effective if you do not want any message loss, but such complicated processes will increase the end-to-end delay. If we can deduct appropriate QoS level utilizing correlation analysis of the relationship between the message loss due to the size of payload and the end-to-end delay, it will be possible to build a optimal service network for push notification services.

Julio analyzed performance of MQTT protocol under OMNeT++ simulation environment[2], which is an open source project and a Network Simulation tool for OMNeT++[3]. From their thesis, he analyzed the performance of several clients along with the number of messages regarding the MQTT protocol, under the simulated environment. Also, the measurement of wireless networks based on a Gilbert-Elliot modeling was tested, but it is hardly reliable to reflect the performance under a real network environment. Our experiment overcomes the limitations of Perez, Julio simulation experiments. We analyzed packets resulted from end-to-end delay and message loss caused by different payloads under different QoS levels of MQTT in real wired/wireless network environments, and drew a conclusion regarding the correlation among such factors.

This paper is organized as follows. In Chapter 2, we will describe the experimental environment and method of measurement. In Chapter 3, based on the experiments conducted in Chapter 2, we will analyze the packets of end-to-end delays and message loss caused by different payloads and QoS levels, and presents the correlation analysis results. And, in the final chapter, Chapter 4, we will present the conclusion of this paper and propose possible future studies.

II. METHODLOGY

In this chapter, we propose a method of measurement for network environment and packets to analyze end-to-end delay, message loss in relation with different QoS Levels and payload sizes. For the experiment, we used Linux-based CentOS 6.3 Final for the server, and an open source project Mosquitto[5] for Broker server software.

MQTT transmits messages through Broker server to form a connection between Publish and Subscribe clients. Figure 2 shows the process of how each Client transmits and receives the message via Broker server according to different Topics.

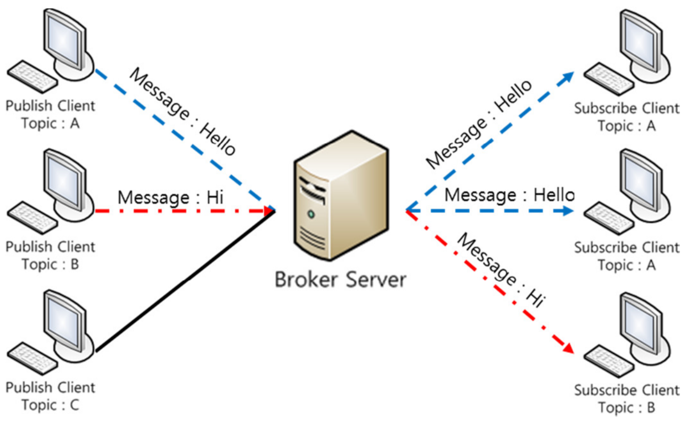

Figure 2. MQTT Message Transmission Process

Publish client notifies the Broker server with topics which it intends to publish. Broker server keeps the published topics, and when Subscribe clients ask for relevant topics, Broker server acts as an intermediary for message transmission. Subscribe clients will be delivered with messages of the topic it asked for from the Broker server.

A. *Wired / Wireless Network Experiment Environment*

To conduct an experiment in a practically real network environment, we have formed the wired/wireless connection between Broker server and Publish/Subscribe clients as shown in Figure 3 below.

To proceed with the experiment in an environment similar to the real network environment, we conducted experiments using a commercial network. Under the customized environment, the reason why we placed clients in external network is to form a practically realistic environment in which Broker server access passes through the commercial network core.

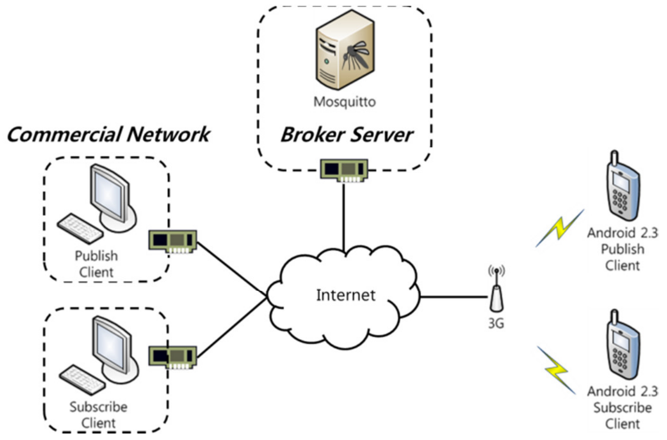

Figure 3. Wired / Wireless Network Configuration

Clients under wireless network environment were tested under Android 2.3 environment. Wireless environment can be considered realistic since the communication from mobile environment goes through 3G network and reaches Broker server.

B. *Packet Collection and Analysis Method*

For wired environment experiment, we used Tcpdump[6] to collect packets. Tcpdump is a tool that can capture packets under the Linux console environment. Tcpdump provides the function which can save the packet into Pcap files. Pcap file is useful for packet analysis when the Wireshark is implemented.

In wireless network environment, we used shark application to collect packets. Since mobile environment sets limitations in collecting packets, we used shark application to capture the packets, save them into Pcap files and analyzed the files with Wireshark.

We analyzed message loss and end-to-end delay by collecting packets between the client and the server during the measurement of 5 minutes. When it comes to end-to-end delay, we used timestamps formed while packets move along from Subscribe server to Publish client via Broker server. We counted the retransmission request packets for the time of five minutes to measure the message loss.

III. EXPERIMENT

In this chapter, we analyzed the message loss and the end-to-end delay in wired / wireless network and presented correlation of message loss and end-to-end delay according to payload and QoS levels.

A. *Wired Network Packet Analysis Result*

Figure 4 below shows the correlation between end-to-end delay in relation to payloads and QoS levels.

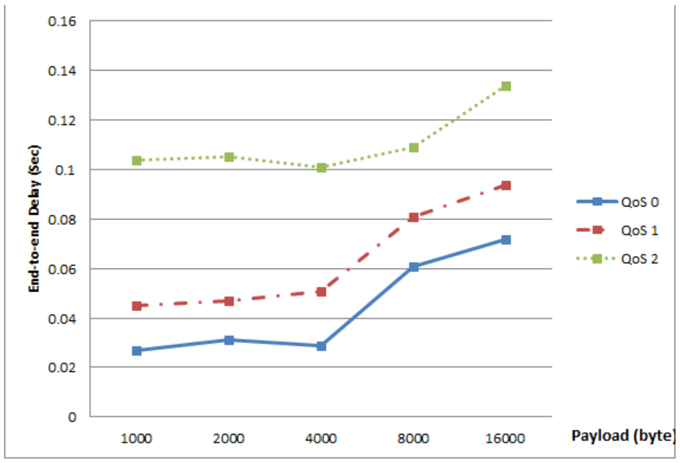

Figure 4. Wired Network Mean End-to-End Delay analysis result

Thus, any message over 4,000 bytes will increase the number of division of packets, and cause longer end-to-end delay. The main reason for QoS 2 to have longer end-to-end delay is the use of 4-way handshake. As a result, QoS 2 does not show a significant change in end-to-end delay for messages smaller than 8,000 bytes. Figure 5 below is a graph that compares message loss in regard to QoS levels and payloads. We were able to confirm that the message loss was reduced by about 1.57 times when using QoS 2 compared to using QoS 0 or 1. This result shows that although the message loss increase with the increase in message size, QoS 2 is able to cut the loss to some degree by using 4-way handshake.

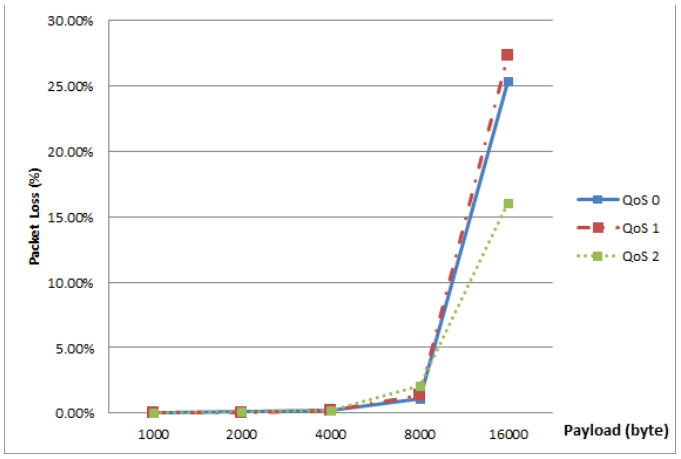

Figure 5. Wired Network message loss analysis result

B. Wireless Network Packet Analysis Result

Compared to those for wired network, the results of measuring end-to-end delay for wireless network showed longer delays. The main contributing factor for such delays is the transmission speed of 3G network. Figure 6 below is a graph which compares the end-to-end delays influenced by payloads and QoS levels in a wireless network environment.

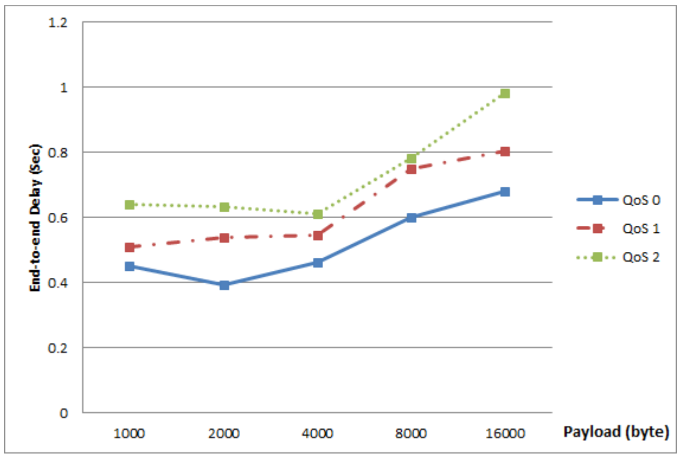

Figure 6. Wireless Network Mean End-to-End Delay analysis result

An experimental result shows that there is a slight difference between the end-to-end delay in wireless network environment and the one in wired network environment. We can confirm that due to the slower transmission speed of the wireless network compared to that of wired network environment, the wireless network tend to have larger end-to-end delay. Analysis of message loss in wireless network environment is shown in Figure 7 below. Although message loss rate is rather stable, experiment results showed that QoS 2 has lower message loss rate than QoS 0 or 1.

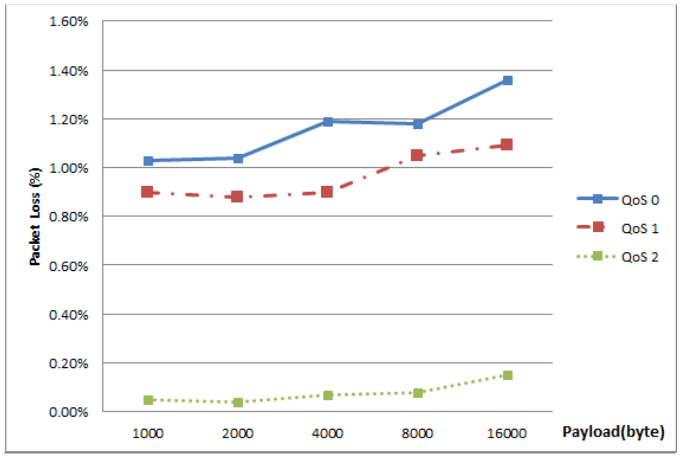

Figure 7. Wireless Network message loss analysis result

C. Analysis correlation about MQTT QoS Level

Based on the results above, we use the correlation of the end-to-end delay and message loss to each QoS level. Correlation is a mathematical model to recognize how variables x and y influence each other. Variables n represents the number of payload measured at each QoS level. Correlation coefficient r satisfies $-1 \leq r \leq 1$, and r value closer to 1 represents a positive correlation. Not to mention, r value closer to -1 represents a negative correlation. In order to analyze the relationship between end-to-end delay and message loss, we use the following correlation formula.

$$r = \frac{\frac{1}{N}\sum_{r=1}^{N}(x_r-\bar{x})(y_r-\bar{y})}{S_x S_y} \quad (1)$$

$\bar{x}, \bar{y}$: The average value of x, y

S_x, S_y : The standard deviation of x, y

TABLE I. ANALYSIS CORRELATION OF MESSAGE LOSS AND END TO END DELAY FOR MQTT QOS LEVEL

	Wired Network	Wireless Network
QoS 0	0.771574	0.878897
QoS 1	0.788814	0.884428
QoS 2	0.991416	0.904699

Table 1 summarizes the correlation between the end-to-end delay and message loss according to different QoS levels. Correlation analysis results in our experiments shows that end-to-end delay and message loss has a strong positive relationship for each QoS level. Thus, we can say that end-to-end delay and message loss is closely related.

IV. CONCLUSION & FUTURE WORK

In this paper, we, the experiments were performed in a more realistic network environment than the existing simulated environment, OMNeT++. Also, we have suggested results of correlation analysis of end-to-end delay and message loss under different QoS levels and payloads. Experiment results suggest that end-to-end delay is significantly associated with message loss under different payloads. If we can utilize MQTT protocol using appropriate QoS level for different payloads, we will be able to build a more effective push notification service network environment. Regarding future studies, we will experiment under various QoS levels and payloads and deduct the most efficient QoS level setting in regard to different payloads. Also, we will analyze MQTT Broker server performance, communication among multiple clients and throughput rates considering different numbers of topics.

ACKNOWLEDGMENT

This research was financially supported by the Ministry of Education, Science Technology (MEST) and National Research Foundation of Korea(NRF) through the Human Resource Training Project for Regional Innovation(2012H1B8A2025942).

REFERENCES

[1] The MQTT protocol, http://www.mqtt.org, cited August, 2012.

[2] P. Julio, "MQTT Performance Analysis with OMNeT++," IBM Zurich Research Laboratory, September, 2005.

[3] A. Varga, OMNeT++, http://www.omnetpp.org, cited August, 2012.

[4] Building Facebook Messenger, http://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920, cited August, 2012.

[5] Mosquitto, http://www.mosquitto.org, cited August, 2012.

[6] Tcpdump, http://www.tcpdump.org, cited August, 2012.

[7] Power Profiling: HTTPS Long Polling vs. MQTT with SSL, on Android, http://stephendnicholas.com/archives/1217, cited October, 2012.

[8] S. Behnel, L. Fiege, G. Muehl, "On Quality of Service and Publish Subscribe," Proceedings of the 26th IEEE International Conference on Distributed Computing Systems Workshops, July, 2006.

[9] K. M. Bell, D. N. Bleau, J. T. Davey, "Push notification service," U.S. Patent 8 064 896, November, 2011.